AD-A044 542 STANFORD RESEARCH INST MENLO PARK CALIF F/G 9/2
HIERARCHICAL REPRESENTATION OF THREE-DIMENSIONAL OBJECTS.(U)
MAR 77 G J AGIN N00014-71-C-0294
UNCLASSIFIED NL

1 OF 1
ADA044542

END
DATE FILMED
10-77
DDC

Final Report *March 1977*

HIERARCHICAL REPRESENTATION OF THREE-DIMENSIONAL OBJECTS

By: GERALD J. AGIN

Prepared for:

OFFICE OF NAVAL RESEARCH
800 NORTH QUINCY STREET
ARLINGTON, VIRGINIA 22117

Contract Monitor: Marvin Denicoff, Program Director
Information Systems Branch

Contract N00014-71-C-0294

SRI Project 1187

Research was sponsored by the Office of Naval Research under Contract N00014-71-C-0294.

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

Final Report

March 1977

46p.

HIERARCHICAL REPRESENTATION OF THREE-DIMENSIONAL OBJECTS

By: GERALD J. AGIN

Prepared for:

OFFICE OF NAVAL RESEARCH
800 NORTH QUINCY STREET
ARLINGTON, VIRGINIA 22117

Contract Monitor: Marvin Denicoff, Program Director
Information Systems Branch

Contract N00014-71-C-0294

SRI Project 1187

Research was sponsored by the Office of Naval Research under Contract N00014-71-C-0294.

Approved by:

PETER E. HART, *Director*
Artificial Intelligence Center

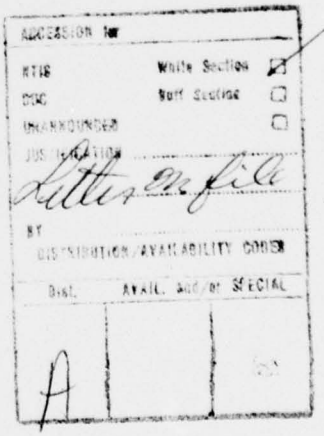
E[illegible]ILE JONES, *Executive Director*
Information Science and Engineering Division

332500 B

ABSTRACT

Many different approaches have been investigated for the computer representation of three-dimensional shapes. Most of them lack characteristics that would facilitate man machine communication using semantic, graphic and visual means This report describes a formal method for describing shape that has desirable characteristics along these lines. The method uses generalized cylinders to characterize the primitives of the representation and hierarchical assemblages of primitives. We describe a set of computer programs that translates formal descriptions of objects into polyhedral models and line drawings.

A scanning range finder is used to obtain three-dimensional information about a scene. Then a set of computer programs characterizes cones, cylinders, and rectangular solids in the image. The method is manually guided, and requires much [illegible] or interaction, but has the potential for extension to a useful sy[illegible] for interactive design, or for formation of the core of a three-dimensional vision system.



CONTENTS

ABSTRACT ii

LIST OF ILLUSTRATIONS iv

I INTRODUCTION 1

II THE REPRESENTATION 4

A. Representation by Spine/Cross Section Methods 4

B. Structural Relationships Among Parts 6
1. General Positional Relations 7
2. Attachment Points 9
3. Snakes and Stacks 10
4. Summary 11

C. Primitives 12

D. Creating Objects from Primitives 13

III THE DISPLAY OF WIRE MODELS 18

IV COMPUTER-AIDED MODELING USING RANGE DATA 21

A. Laser Range Finder 22

B. Operations with Masks 25

C. Surface Fitting 33

D. Interactive Model Building 37

REFERENCES 40

ILLUSTRATIONS

1. Model of a Screwdriver 5

2. Model of an Airplane 5

3. Model of a Chair 5

4. A One-Inch Cube on Top of a Two Inch Cube 9

5. DIMENSIONS Property for SCREWDRIVER 15

6. COORDINATE-FRAME Property for SCREWDRIVER 16

7. PARTS Property for SCREWDRIVER 17

8. STRUCTURE Property for SCREWDRIVER 17

9. Indoor Scene Chair on Platform 23

10. Range and Intensity Images for Chair Scene 23

11. Range and Intensity Images for a Test Scene 26

12. Mask A: Jump Boundaries 28

13. Outlining a Shape 28

14. Mask B Outlined Area 28

15. Histogram of y Values 28

16. Mask C: Points Selected from Histogram 29

17. Rectangle Outlined with Cursor 29

18. Mask D Generated from Outline 29

19. Mask E: Masks C and D ORed 29

20. Range and Intensity Images for Airplane Scene 31

21. Mask A: Jump Boundaries 31

22. Mask B: Inverse of Mask A 31

23. Mask C: Isolated by Connectivity Analysis 32

24. Mask D: Growing 32

25. Histogram of z Values 32

26. Mask E: Points Selected from Histogram 32

27. Fitted Cylinder and Jump Boundaries 36

28. Top View of Fitted Cylinder and Range Data 36

29. Range and Intensity Images for Model Airplane Scene . . . 37

30. Primitive Surfaces Fit to Range Data 39

31. Primitive Surfaces Fit to Range Data: Alternate Viewpoint 39

I INTRODUCTION

The computer representation of three-dimensional shape has occupied the attention of a number of researchers in the past several years. This is so at least partly because a useful theory of shape would have applicability to a wide variety of fields, such as design automation, manufacturing automation terrain mapping, vehicle guidance, archaeology, restoration of works of art, surveillance, and intelligent robots in general. And aside from any practical applications, the problem has considerable scientific and mathematical interest.

Finding a useful and general method for talking about shape is not a simple problem. Methods that are primarily numerical are usually limited in generality. Humans have great difficulty describing shapes to one another unless the shapes are familiar ones; the verbal description of a face, for example, is rather difficult.

Various systems have evolved or have been invented for specific tasks. Generally, they fall into one of two categories: surface representations and volume representations.

Surface may be approximated by triangular patches [1] with various interpolation schemes. Rectangular patches have been used successfully with spline interpolation for the precise specification of airfoils and automobile bodies [2]. Contour maps are widely used for a number of applications.

In volumetric modeling, the usual approach is to represent objects as intersections and unions of simpler objects. Combining polyhedra and cylinders in this way has been shown to be useful in the description of machined metal parts for design automation [3-7]. Generalized cylinders have been used for modeling many everyday shapes for computer vision [8,9]. A more specialized use of generalized cylinders has been used in classification of pottery shapes [10].

The facet of the problem we are most interested in is the representation of shape in ways that can be easily communicated and understood by humans This rules out methods that involve equations or large arrays of numbers. Natural media of communication include words, pictures, and examples. We use the terms semantic communication, graphical communication, and visual communication to denote these three modes.

Informal studies show that notions of similarity and difference are important to human communication of shape concepts. The descriptions used frequently begin with a familiar object, then mention significant differences between the object being described and the familiar one. Many alternate descriptions of the same object are possible.

This report describes one approach to describing shapes in natural, human terms. The method is hierarchical, using generalized cylinders to characterize primitive elements and their assemblies into higher level subparts and parts.

Section II of this report describes the semantic basis of our system: a formalism for describing shape. This formalism makes use of a syntax based on the programming language LISP, but the principles of the representation should be translatable into other languages with suitable symbolic capabilities.

In Section III we give details of the graphic component: a set of computer programs that interpret descriptions in our descriptive formalism to produce polyhedral models. The "wire frame" of the polyhedral models may be displayed in perspective on a graphics terminal. This program, when used in conjunction with facilities to edit the semantic shape descriptions, constitutes a very rudimentary interactive design system.

A visual capability is described in Section IV. A set of computer programs deals with data obtained from a time-of-flight range finder, to yield descriptions of primitive cylinders, cones, and rectangular solids in the scene. The system is manually guided and

presently requires considerable operator interaction. With some relatively straightforward extensions it should become a reliable part of a system to teach shape descriptions by example and it could also form the basis for an automatic computer vision program.

II THE REPRESENTATION

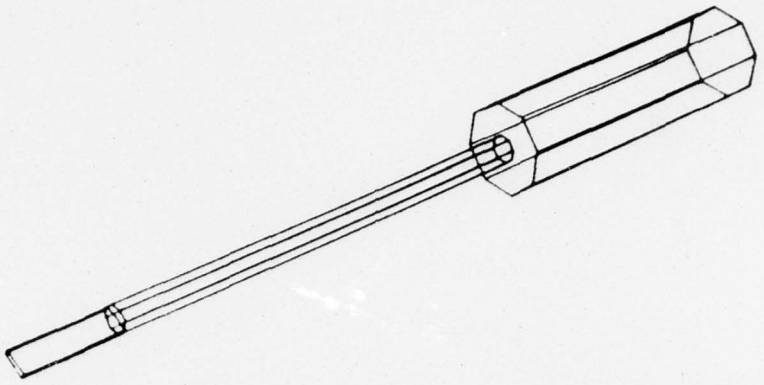
We have developed a representation based on a spine/cross-section method. The primitives of the representation are generalized cylinders, comprised of a central axis or spine, and a cross section function defined on that axis. The primitives may be combined in ways that make use of the axes inherent in the primitives.

In many cases, however, spine/cross-section representation is not appropriate, and more general but less intuitive methods are needed We provide the capability for representing arbitrary spatial relationships for these cases.

Examples of some of the objects that may be usefully represented by our methods are shown in Figures 1, 2, and 3. Figure 1 shows a screwdriver, for which spine/cross-section methods are completely sufficient. Figure 2 is a representation of an airplane. Spine/cross-section methods are used to define most of its structure and shape, but some auxiliary positioning methods are needed for some of its component parts. For the chair of Figure 3, only general structural relationships will conveniently work

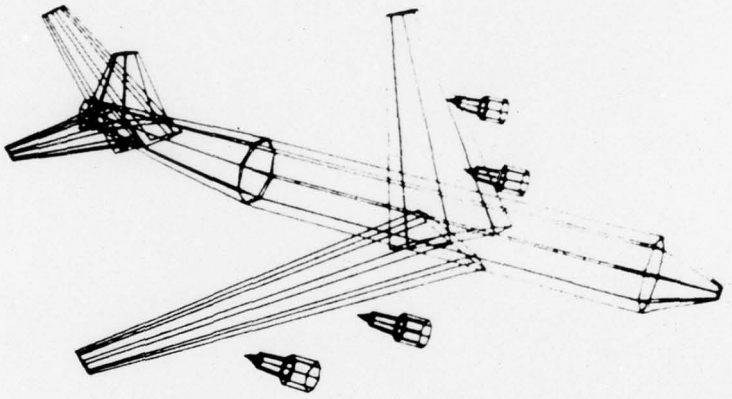
A. Representation by Spine/Cross-Section Methods

Representation by spine/cross-section methods uses a space curve, or axis, and a cross section function defined on this axis. The primitives of this representation have been called generalized cylinders [8]. Given a simple object, We may determine its description by locating an axis such that the object's cross section (normal to the axis) varies in a uniform manner along the axis. A description of a complex object may be built up by "cutting and pasting" the descriptions of its constituent parts.



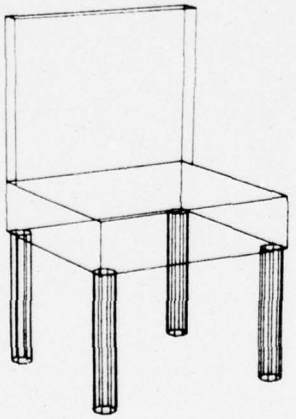
SA-1187-25

FIGURE 1 MODEL OF A SCREWDRIVER



SA-1187-26

FIGURE 2 MODEL OF AN AIRPLANE



SA-1187-27

FIGURE 3 MODEL OF A CHAIR

These concepts provide a natural, intuitive way of representing solid objects. The primitives of such a model represent portions of solid objects, instead of surfaces or appearances. The method allows segmentation of a complex object into parts easily represented and a hierarchical approach to the description of objects. Representation by generalized cylinders is synthetic in nature; that is given a model one can uniquely synthesize the contours of the object.

As an analytic representation (for generating a model to describe a given real object), it is easy to describe any solid object that has a straight axis. But in general, analysis in terms of an axis and cross section does not yield a unique answer; heuristic or interactive techniques are needed to select natural or useful axes.

Segmentation is a critical issue here. For analysis or for synthesis it is necessary to find parts of an object that may be described simply and to "paste" these segments together. (In the pasting process, the individual segments may interpenetrate.)

In our representation, objects may be composed of an assemblage of smaller subobjects, or they may be primitive. Only primitive objects have an explicit cross section description; for all other objects the shape is described by the shapes of and relationships among its component parts. Hierarchical representations of complex objects may be built up by independently describing subparts, then describing the structural relationships among the subparts.

B Structural Relationships Among Parts

There are three fundamental ways the relationships among parts may be specified in our representation: by snakes, by attachment points and by arbitrary displacements and rotations. Usually any object of significant complexity will use all three of these methods for its elucidation.

Structural relationships involving snakes describe objects displaced along a single axis, like beads on a string. The axes of the

individual parts combine to form the axis of the assembly. Relationships involving attachment points are like Tinkertoys: parts have predefined points at which other pieces may be attached. Displacement by arbitrary transforms is the most general of the three methods; the other two can be considered special cases of arbitrary displacements. But snakes and attachment points are closer to intuitive notions of structure, and usually assume a more compact form.

For ease in exposition, we will consider general relationships first, then return later to the easier-to-use structures.

1. General Positional Relations

Syntactically, a general structural relationship is a list of pieces and transforms:

```
gen-structure ::= NIL
              ::= ( transform . gen-structure )
              ::= ( piece . gen-structure )
```

(We will indicate the syntax of structural relationships by productions similar to the above. Words in upper case denote specific literal atoms, i.e., themselves. Words in lower case denote S-expressions that are defined or described elsewhere. For a description of dotted pairs and lists see Weissman's introduction to LISP [11].)

A transform is an operation that changes one coordinate system into another. Transforms in a structure description can describe positional relationships as well as scale or shape changes.

The two most frequently used primitive transforms are for rotation and translation (rotary and linear motion):

```
transform ::= ( TRANSLATE direction number )
          ::= ( TRANSLATE number number number )
          ::= ( ROTATE direction number )

direction ::= X | +X | -X | Y | +Y | -Y | Z | +Z | -Z
```

The first form of the TRANSLATE transform generates a motion parallel to a single axis. For the second form of TRANSLATE, the three numbers indicate simultaneous motions in x, y, and z, respectively. The ROTATE transform generates a rotation about one axis.

Several additional transformation primitives give additional descriptive power to the representation:

```
transform ::= ( SCALE direction number )
          ::= ( SCALE number number number )
          ::= ( MIRROR direction )
          ::= ( SKEW direction number )
```

A piece may be a symbolically named subpart (see Section D) or it may be another structure

```
piece ::= part-name
      ::= structure
```

This allows a hierarchy of structures within any level of the hierarchy of parts and subparts.

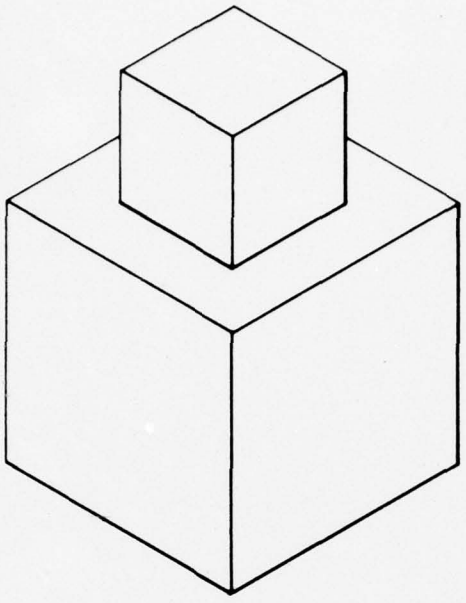
A general structure descriptor should be interpreted from head to tail. The position of any piece with respect to the coordinate system in which the overall structure is described is the product of all the transforms that precede the piece in the list. The structure descriptor may be thought of as instructions to maneuver a "bus" that drops pieces at its local origin of coordinates between maneuvers.

Consider the task of specifying a structure in which a one-inch cube rests on top of a two-inch cube, as shown in Figure 4. Suppose the two subparts have been defined as will be shown in Section D and given the names CUBE1 and CUBE2. Then the structure may be described by the following S-expression:

```
( CUBE2 (TRANSLATE +Z 2) CUBE1 )
```

We interpret this description by placing the two-inch cube at the origin of the local coordinate system. The coordinate system is then translated two inches upward, and the one-inch cube is placed at the relocated origin of coordinates.

The chair of Figure 3 was assembled from primitive elements using only translational and rotational transforms.



SA-1187-28

FIGURE 4 A ONE-INCH CUBE ON TOP OF A TWO-INCH CUBE

2. Attachment Points

A difficulty in the preceding example is that we must know the height of CUBE2 in order to say where to place CUBE1. The situation might be somewhat easier if symbolic names were used instead of numbers, but there still is the problem of ascertaining that the names match. It would be much more useful to have the capability of saying, in effect, "place CUBE1 on top of CUBE2."

Attachment points provide one way to accomplish this. For all primitive objects the attachment points TOP, SIDE, BACK, and BASE are predeclared; for nonprimitive objects, attachment points may be declared as we shall see below. These are named places on the surface of the object on which (or relative to which) to locate other objects.

Syntactically, an attachment point reference is legal anywhere a transform is legal; thus attachment point references are included in the productions for transforms:

transform ::= (ATTACH part-name attachment-point-name)

The "part-name" of the reference is the symbolic name of another subpart at the appropriate level of hierarchy. Naturally, the position of the part must be known earlier in the interpretation of the structure.

Recasting our example of two cubes in terms of attachment points, we find that the structure of Figure 4 becomes

(CUBE2 (ATTACH CUBE2 TOP) CUBE1)

An attachment point is carried in the data structure as a transform relative to the BASE of an object, i.e., the origin of the coordinate system in which the object is described. For nonprimitive objects, new attachment points may be specified by a name and one or more transforms to specify the attachment point location (see the ATTACHMENTPOINTS property, Section D).

Attachment points are particularly useful in describing biological (animal, human) shapes and a wide variety of manufactured objects. The air-compressor model reported in last year's report [12] was modeled almost entirely with attachment point structures.

To represent the airplane of Figure 2, we used attachment points to designate the attachment of the wings and tail assembly to the fuselage.

3. Snakes and Stacks

A snake is a group of objects sharing a single extended axis. It is the snake relationship that gives our representation its spine/cross-section capabilities. All primitive objects are defined to have an axis and a length. Nonprimitive objects may also have axes and lengths, either explicitly specified or implicitly derived. To assemble parts in a snake structure we place the components end-to-end with their axes touching, like stringing beads on a necklace.

In our implementation, snakes are created with the stack construct. As a LISP S-expression, a stack is a list containing the atom STACK followed by an arbitrary number of pieces:

stack ::= (STACK piece piece ... piece)

Each piece must have an axis and length defined. The structure created by interpreting this expression will have its axis vertical with pieces stacked from bottom to top. If the ultimate orientation of the structure is to be other than vertical, additional transforms will be needed.

Continuing our example of Figure 4, we have the following S-expression as a representation of the structure using a stack:

(STACK CUBE2 CUBE1)

The screwdriver of Figure 1 was modeled exclusively with the STACK construct.

Axes with bends or corners in them may be created by extending the syntax of the stack primitive to allow rotation transforms between the component parts. Then the syntax becomes:

```
stack ::= ( STACK . stacktail )
stacktail ::= NIL
          ::= ( piece . stacktail )
          ::= ( rotation-transform . stacktail )
```

Each primitive object has a predefined axis and length. When parts are assembled in a stack, the combined axis and length are assigned to the structure. For nonprimitive objects assembled by other means, an axis may be defined as a length followed by zero or more transforms that will make the axis vertical and resting on the origin of coordinates (see the AXIS property, Section D).

4. Summary

For the example of the two blocks stacked on top of one another, the three methods of specifying their relative positions achieved the same result by using the same interpretive machinery. But

the compactness of the description varied, as did the amount of variation allowable in the description. Description by means of stacks is the most compact and most intuitive; it requires that parts to be mated have their axes lined up. Description of structure by means of attachment points removes the restriction of coincident axes, but requires an explicit description of the attachment point in cases where the attachment point is not predefined. Description by means of arbitrary transforms is the most general of all, but does not easily allow displacements to be a function of the size of the objects to be used as a base.

C. Primitives

At the lowest level of the hierarchy of parts and subparts are the primitives, pieces that are not further divisible and about which the drawing routines have specific knowledge.

The most basic primitive we provide is the cylinder. Specifically, we mean a right circular cylinder; the ends are square and the cross section is circular. It is described in its canonical orientation: with its axis extending upward (in the +z direction) from the origin of coordinates. The two specifiable dimensions are LENGTH and DIAMETER.

The other primitive we provide is the brick. A brick is similar to a cylinder in all respects except the shape of its cross section. Specifiable dimensions are LENGTH (parallel to the z-axis in the canonical orientation), WIDTH (parallel to the x-axis), and DEPTH (parallel to the y-axis).

It would be easy to create other primitives with new cross section shapes. But the cylinder and the brick have been adequate so far for our purposes. Clearly a general cross section description facility would be useful for describing such things as rulers or fluted screwdriver handles. But we have not given much thought as to how to implement such a capability.

Note that some shapes are derivable from the circle and the rectangle. For example, an elliptical cross section may be generated by transforming a cylinder with a nonuniform SCALE transform.

Cross section dimensions are allowed to vary linearly from one end of the segment to the other. This allows the generation of conical or pyramidal shapes. The DIAMETER or DEPTH and WIDTH given apply to the base of the primitive. The dimensions at the top are determined by the specifiable parameter TAPER. A TAPER of zero (the default) implies a uniform cross section; otherwise the top dimensions are (TAPER+1) times the bottom dimensions. A TAPER of -1 yields a cone or pyramid with a point at the top.

The only type of axis segment we currently permit is a straight one. Curved axes may be approximated by piecewise linear axes by including rotation transforms in a STACK construct. The obvious first-order improvement to the current state of affairs would be to allow axes to have a curvature (numerically, the reciprocal of the radius of curvature). The plane of curvature would have to be fixed with respect to the canonical position. This first-order extension should be able to handle 95% of the usual cases of objects modeled with curved axes.

D. Creating Objects from Primitives

There are three classes of object in our representation: prototypes, descriptions, and instances. A prototype contains information regarding a class of objects, including nominal or typical dimensions, and allowable variation. A prototype is created by a programmer, who decides how to represent a particular object or piece, or by a program that decides how with interactive help from a human teacher.

Descriptions and instances refer to particular objects. A description is a concise recipe for copying a prototype and assigning specific values to its variables to form an instance. The instance is a copy of its prototype description, with all of its parameters,

dimensions, and options given specific values. The process of converting a description to an instance is called instantiation.

For example, the prototype SCREWDRIVER contains all known information about screwdrivers in general. If we wish to talk about a screwdriver 14 inches long and with a narrow blade, we would construct the description

```
(SCREWDRIVER (LENGTH 14)
             (BLADE-THICKNESS (TIMES DEFAULT .5) ) )
```

Upon instantiation, the description might produce an instance whose name is SCREWDRIVER0023.

Prototypes and instances are carried in our representation as LISP atoms with property lists. Descriptions are S-expressions similar to the one above. The principal entries on the property list of a prototype are as follows:

DIMENSIONS names the sizes, angles, and other parameters that may vary or be specified in a description, and their defaults.

COORDINATE-FRAME may define a local coordinate system.

PARTS specifies the pieces to be assembled to produce this object.

STRUCTURE gives the spatial relationships of the parts to each other and to the whole, as described in Section B.

ATTACHMENTPOINTS can provide named locations for assembly with other objects.

AXIS specifies the direction and length of the central axis of the object.

Instantiating an object involves creating a context in which symbolic expressions may be evaluated. (The spaghetti stack capability of INTERLISP permits the preserving of contexts and their variable bindings.) A tree of contexts parallels the tree of subparts.

When an instance is created from a prototype, a new context is created. The DIMENSIONS and the COORDINATE-FRAME properties of a part provide a list of names that will be bound in the new context. In other words, if the variable name LENGTH is part of the DIMENSIONS property, a

new variable of that name will be created as a part of the new context. Usually each variable name will have a default value or expression for the variable to take. The defaults may be overridden by modifiers in a description (as in the case of the 14-inch screwdriver, above).

```
(PUTPROPS SCREWDRIVER DIMENSIONS
    ((LENGTH (BETWEEN 6 24))
     (HANDLE-LENGTH (TIMES (BETWEEN .2 .5) LENGTH))
     (SHAFT-LENGTH (DIFFERENCE LENGTH HANDLE-LENGTH))
     (SHAFT-DIAMETER (TIMES LENGTH (BETWEEN .01 .05)))
     (TIP-LENGTH (TIMES SHAFT-DIAMETER 4))
     (HANDLE-DIAMETER (TIMES LENGTH (ABOUT .1)))
     (BLADE-THICKNESS (TIMES SHAFT-DIAMETER
                             (BETWEEN .1 .3)))
     (BLADE-LENGTH SHAFT-DIAMETER)))
```

SA-1187-29

FIGURE 5 DIMENSIONS PROPERTY FOR SCREWDRIVER

Figure 5 lists the complete DIMENSIONS property from our prototype of a screwdriver shown in Figure 1. The large number of dimensions is necessary to be able to precisely specify the shape for drawing routines, but the programmer who makes use of the prototype description need not concern himself with the ones he is not explicitly modifying.

The DIMENSIONS property of the chair in Figure 3 contains the dimension names SEAT-WIDTH, SEAT-DEPTH, and LEG-DIAMETER (among others). The numbers corresponding to these names are used to calculate the relative positions of the legs with respect the the chair's seat.

At some time or another, each dimension must be evaluated, i.e., a number must be calculated and assigned. At instantiation time, we have the choice of copying the unevaluated expressions from the property list of the prototype to the variable bindings of the new context, or of evaluating each expression and storing the number as the value of the variable. If expressions are stored, evaluation will take place only when a number is needed (for instance by the drawing program). This keeps useful information available for question answering purposes, and will permit changing top-level parameters and positions without the necessity of modifying the data structure. On the other hand, the

storing of numbers is vastly more efficient if semantic capabilities are not needed. In our implementation of the representation, provision is made to carry out instantiation in either mode: numeric or symbolic.

A unique feature of our representation is the ability to specify local coordinate directions. The need for this becomes apparent when assembling parts into a major assembly. It is frequently advantageous to describe a part in one orientation when its eventual mounting will be in another direction. Consider, for example, what it means to refer to the "top" of a bolt. If the bolt were considered in isolation, the top would be assumed to refer to the head end, but if the screw were inserted in a horizontal position the "top" would be meaningless. We permit specifying direction names such as TOWARD-HEAD or TOWARD-TIP that are "embedded" in the object regardless of the object's eventual orientation. Some more useful applications include establishing a FRONT and BACK to assemblies, or a BOW, STERN, PORT, and STARBOARD for ships, aircraft, and vehicles.

```
(PUTPROPS SCREWDRIVER COORDINATE-FRAME)
    ((HANDLE -Z)
     (TIP Z)))
```

SA-1187-30

FIGURE 6 COORDINATE-FRAME PROPERTY FOR SCREWDRIVER

Figure 6 is the COORDINATE-FRAME property of the screwdriver. The entries in this list are treated identically to the variables of the DIMENSIONS property.

Every nonprimitive object should have a PARTS property. Each part is given a symbolic name that will be referred to in the STRUCTURE property and is described by a part description with appropriate modifiers. Instantiating any object causes its parts to be instantiated also, so that a hierarchy of instances will parallel the hierarchy of prototypes.

Figure 7 is the parts list for the screwdriver. The prototype SCREWDRIVER-HANDLE is also a component of other parts; the parts lists

```
(PUTPROPS SCREWDRIVER PARTS
    ((HANDLE (SCREWDRIVER-HANDLE (LENGTH HANDLE-LENGTH)
                                 (DIAMETER HANDLE-DIAMETER)))
     (SHAFT (CYLINDER (LENGTH (DIFFERENCE SHAFT-LENGTH
                                          TIP LENGTH))
                      (DIAMETER SHAFT-DIAMETER)))
     (BLADE (BRICK (LENGTH SHAFT-DIAMETER)
                   (WIDTH BLADE-THICKNESS)
                   (HEIGHT TIP-LENGTH)))))
```

SA-1187-31

FIGURE 7 PARTS PROPERTY FOR SCREWDRIVER

of other objects point to it. The modifiers in the descriptions refer to the variables bound in accordance with the DIMENSIONS property.

```
(PUTPROPS SCREWDRIVER STRUCTURE
    ((STACK HANDLE SHAFT BLADE)))
```

SA-1187-32

FIGURE 8 STRUCTURE PROPERTY FOR SCREWDRIVER

STRUCTURE, AXIS, and ATTACHMENTPOINTS properties have been explained in the preceding sections. Figure 8 shows the STRUCTURE property of the screwdriver. The AXIS property of any instance can be computed from the axes of the three parts, because the STACK construct preserves the axis. The screwdriver has no attachment points defined.

III THE DISPLAY OF WIRE MODELS

We have a set of techniques and computer programs that will draw simple "wire frame" displays of parts described in our formal representation. We do this not only to demonstrate that graphical interaction with the models is possible, but also to ensure that our modeling techniques are correct and unambiguous. Unless we are capable of displaying a number of parts in correct relationship to one another, we cannot be sure the relationship has been adequately defined. A graphical capability is necessary for debugging the individual models, and for assessing the adequacy of the descriptive schema itself.

It is not our intention to invent new techniques for the display of polyhedral models. Many other systems exist for display of polyhedra [3-7]: these systems offer advanced capabilities such as hidden-line elimination, shading, and calculation of interpenetration. There is sufficient information available in our models to interface with any of these programs.

Once an object has been instantiated, the instance may be passed to the drawing routines for display. This process occurs in two stages. First, the spine/cross-section representations are converted to approximating polyhedra, and the corners and edges are stored in a buffer array. Then the picture is drawn on the display console, using a perspective transformation whose parameters are controllable by an operator at the terminal console.

The display subroutine recursively examines successively smaller and smaller elements of an object or assembly, until it reaches the terminal nodes of the hierarchy, the primitive objects. Each primitive object has its own drawing routine; the cylinder drawing routine is typical. To draw a cylinder requires knowing its dimensions (LENGTH and

DIAMETER) and its position in space. The atoms LENGTH and DIAMETER are evaluated in the context of the variable bindings of this cylinder. If numeric information has been bound in the context of the part, the evaluation will yield a direct answer. Otherwise, repetitive evaluation must occur until a numeric answer is found.

The "evaluation" of a position involves a more complicated set of actions. The position of a top-level object must be specified at the time it is instantiated. The position of each subpart of that object is the product of two transforms: the position of its parent part (the next higher level in the hierarchy) and its relative position within the parent assembly. The position within the parent may be computed from the STRUCTURE property, and it is stored in the property list under the property POSITIONINPARENT. The position of any low-level subpart instance may be computed as the product of its POSITIONINPARENT property, times the POSITIONINPARENT properties of all higher subassemblies in the hierarchy, times the position of the top-level assembly.

All the routines discussed so far are coded in INTERLISP. This fits well with the symbolic nature of the data we are handling. But for the actual display of "wire frame" models, and for the analysis of range data, a more numerically oriented language is necessary. Consequently we have two cooperating programs in separate forks of the TOPS-20 operating system: the LISP fork acting as a top level control for the SAIL fork.

Transforms (positions and orientation descriptors) have been carried in symbolic form up to now, as S-expressions of the forms (TRANSLATE ...) and (ROTATE ...). Symbolic multiplication of these transforms is performed as described in last year's report [12]. Numerical information will be needed for the actual display, so an evaluation routine generates a 4 x 4 homogeneous transform matrix according to the "directions" in the symbolic transform [13]. The dimensions and transform are now ready to be passed to the actual drawing routine.

Cylinders are approximated by octagonal prisms for display purposes. To display any polyhedron, the three-dimensional coordinates of each of its edges are computed and stored in a separate display data structure. When the edges are plotted in perspective on the display screen, the polyhedra appear transparent, with wire edges. No attempt at hidden line elimination is made.

The edges are drawn according to a perspective transformation representing an imaginary camera. The position, orientation, and internal parameters of this camera are controllable by a keyboard interpreter to produce an arbitrary view of the object or assembly. Single-letter keyboard commands simulate the translation or rotation of either the scene or the camera with respect to a variety of coordinate systems. Split-screen stereo may also be produced.

When a polyhedron is placed in the display data structure, the indices of the first and last points in that data structure are stored in its property list so that if changes are made to the part's description (i.e., if the part is moved) the display data structure may be updated without redrawing the entire scene.

IV COMPUTER-AIDED MODELING USING RANGE DATA

This section describes techniques for deriving models of actual objects. Depth information is extracted from real-world scenes using a scanning laser range finder. The resulting data are treated as a two-dimensional array of three-dimensional coordinates. By a sequence of interactive steps, patches from this image are isolated and fitted with surfaces corresponding to the primitives of our spine/cross-section models.

The techniques we describe can form the basis for an interactive system for mechanical design, or for describing new objects to an intelligent computer program that models shape for pattern recognition. What we have accomplished is not by itself computer vision, but our methods can form the basis for a computer vision system. The principal difference between interactive modeling and computer vision lies in where the intelligence and decision making reside. At present, a human operator decides what operations are to be performed and in what order. To give this capability to a machine requires that it have the ability to perceive an overall goal, to measure progress toward that goal, and to predict the likely consequences of actions. The core of basic actions we have implemented can constitute the low-level actions available to such a system.

A. Laser Range Finder

An SRI-developed experimental instrument producing registered range and intensity data is used to derive range data for our purposes [14]. Light from a 15-mw laser is modulated with a 9-MHz sinusoidal signal and aimed at an object. Reflected light is detected by a photomultiplier and is phase-compared with a reference beam. Since the phase shift of the detected signal is proportional to the time of flight of light from the scanning apparatus to the object and back again, the phase shift is a measure of range.

A pair of mechanical scanning mirrors allows the beam to scan an indoor scene. The accuracy of the instrument is about one centimeter rms or better. Unfortunately, its speed is not great, several hours being required to scan a scene of 128 x 128 points. A particularly useful feature of this device is that it produces both range and intensity (brightness) for each point.

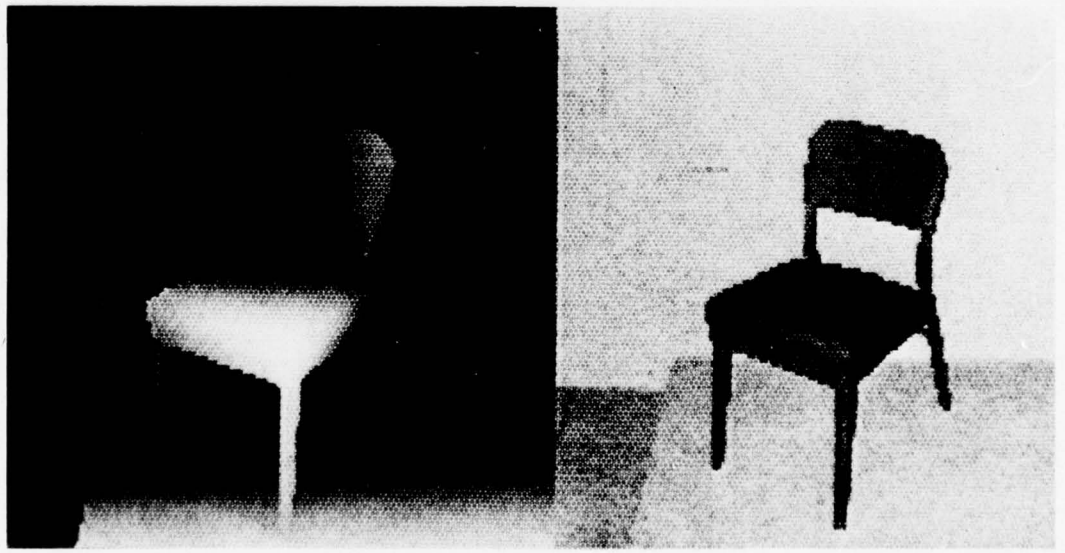
Figure 9 is a photograph of an indoor scene consisting of a chair resting on a low platform. Figure 10 displays the range and intensity data derived from scanning the scene with our range finder. For the left half of Figure 10, range values have been converted to six bits of brightness for display purposes. In the right half, the intensity for each pixel has been multiplied by the square of its range to compensate for the reduction of brightness of far points.

To convert range readings to three-dimensional Cartesian coordinates, we make use of homogeneous coordinate transforms [13]. We assume that the geometry of the situation is similar to the taking of a picture with a pinhole camera. This assumption is not strictly true since at least one of the mirrors sweeps equal angles rather than equal linear distances at the image plane. However, the assumption is accurate enough for our purposes.

The procedure used to obtain x, y, and z for a single range reading is first to convert the reading from the phase meter to inches, next to change the "slant range" thus obtained to an equivalent "depth," and

SA-1187-33

FIGURE 9 INDOOR SCENE: CHAIR ON PLATFORM



SA-1187-34

FIGURE 10 RANGE AND INTENSITY IMAGES FOR CHAIR SCENE

finally to transform from ranger image coordinates to Cartesian coordinates via a homogeneous coordinate transformation.

The homogeneous transform for converting image coordinates to Cartesian coordinates may be represented by a matrix multiplication as follows:

$$\begin{bmatrix} x * H \\ y * H \\ z * H \\ H \end{bmatrix} = [\,M\,] * \begin{bmatrix} I \\ J \\ 1 \,/\, D \\ 1 \end{bmatrix}$$

where I and J are the horizontal and vertical image coordinates and D is the "depth" (to be described below). M is a 4 x 4 homogeneous transform matrix obtained by multiplying together the following transforms: a translation toward the "lens center" of the laser scanner, three rotations corresponding to its orientation, and a projective transform involving separate scale factors for the horizontal and vertical directions. The result on the left-hand side of the equation should be divided by its last element (the scalar H) to obtain the actual coordinates.

The depth D must be measured parallel to the principal ray of the ranger's coordinate system. This may be found by solving for one side of a rectilinear solid where the main diagonal is the measured range and where the other two sides are determined from the I and J scan coordinates and the mirror deflection constants.

A calibration procedure is necessary to establish the correct constants for the complete conversion from raw phase meter units to Cartesian coordinates. (The origin of coordinates is designated by a mark on the floor.) Calibration involves several phases.

The first phase involves measuring, with tape measure and plumb bob, the position of the center of the deflection mirror with respect to our absolute coordinate system. This is a delicate and time-consuming task, but once done it need not be repeated so long as the mounting of the range finder is not moved.

The second phase has as its object the determination of the orientation of the scanner, and of its vertical and horizontal deflection constants. The basic procedure follows the outlines of that used by Sobel to calibrate a television camera [15]. The laser is aimed at several points in the field of view. The image coordinates used to deflect the laser spot are recorded, and the physical coordinates of the laser spot are measured. (Pieces of graph paper taped to the wall are useful in this measurement.) If we have a set of hypothetical values for the rotations and deflection constants that go into the calculation of the matrix M, the image coordinates of the spot may be calculated. These coordinates will not, in general, be the same as the image coordinates recorded. A hill-climbing routine finds the values of the rotations and deflection constants that produce the best agreement between predicted and measured image coordinates of the points.

The final phase uses the data from the previous phase to parameterize a linear relationship between arbitrary units of the phase meter output and inches of range.

B. Operations with Masks

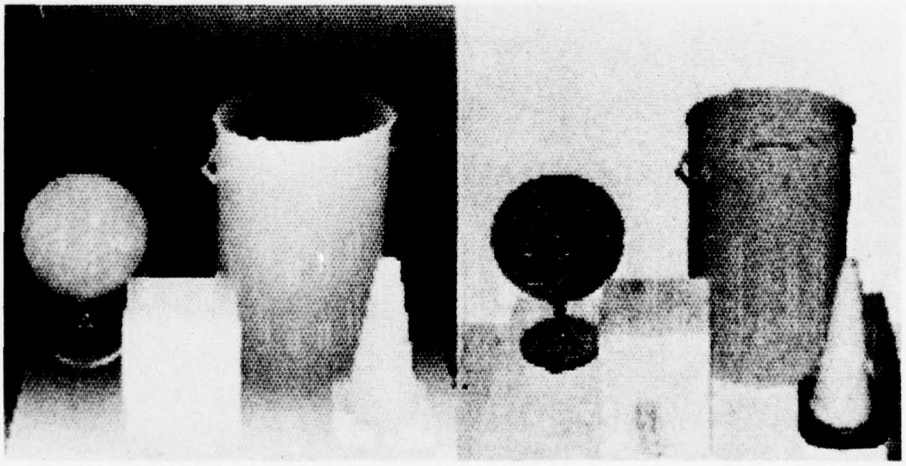
Several steps are necessary to fit models of objects to range data. The first step is isolation of the relevant data from the image. Then follows segmentation of the isolated data into parts, each to be represented by primitive snakes or higher-level assemblies. Next we have fitting of primitive surfaces to the range data. Finally, there must be an assembly of the component parts into a whole that is recognizable as an example of a prototype from the universe of known objects.

These are not, in general, easy tasks to perform automatically. Attempts have been made to perform the four steps in a bottom-up fashion, with limited success [8,9]. For a more robust approach, the various levels must interact, with failure at one level guiding retries at the lower level. Finding a mechanism for this interaction constitutes a continuing common theme for all vision research.

Rather than attack the problem head-on, we have selected a phased approach that starts out with a set of low-level routines for dealing with range data and with the correspondence between range data and models, and uses manual direction to sequence operations and select parameters. The low-level routines are the components of an eventual system that will perform automatically.

The isolation and segmentation operations on pictures are implemented by generating and manipulating binary masks. For any operation on range data, a mask usually specifies which data points are to be considered. Up to 26 masks may be generated, each containing one bit per pixel of the range data (usually 128 x 128).

An executive program allows interactive display and selection of operations on masks. Operations include detection of jump boundaries in the range data, logical operations (AND, OR, NOT), windowing, histogramming, connectivity analysis, outlining, growing and shrinking operations, projective display of selected points, and the reading and writing of disk files.



SA-1187-35

FIGURE 11 RANGE AND INTENSITY IMAGES FOR A TEST SCENE

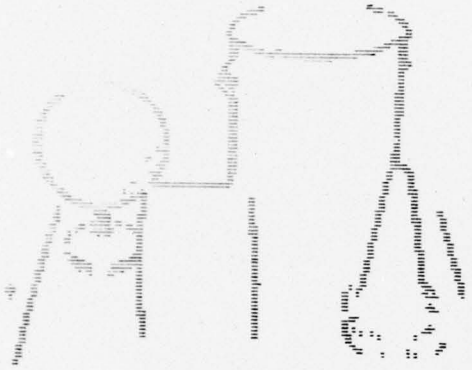
An example will illustrate the use of masks. Figure 11 displays the range and amplitude obtained from a test scene consisting of a cylindrical trash container, a globe, a plastic cone of the kind used to mark wet floors, and a rectangular box. We wish to isolate the trash container for analysis by the cylinder fitter.

To start the analysis, we first find the jump boundaries in the range picture. Every point in the scene that has a neighbor with a range differing from its own by more than 3 inches has its corresponding bit turned on in the mask. The result is a crude outlining of the major objects in the scene. The result of jump boundary detection is shown in Figure 12.

The next step is to use a cursor on the display to outline the area we are interested in. The outlining need not be exact; subsequent steps will eliminate those points we are not interested in. The boundary we draw is large enough to include the entire visible portion of the waste container and a small margin beyond. Figure 13 shows the outline drawn with the cursor and Figure 14 shows the mask generated as the interior of the outline.

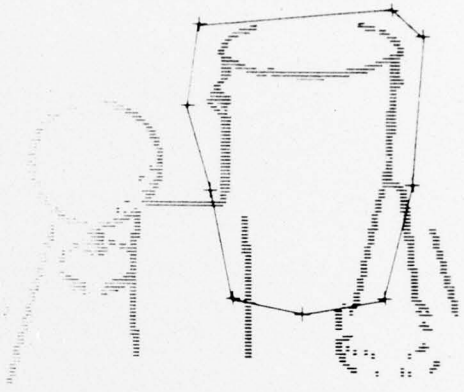
A histogram filtering procedure provides further refinement to the mask. Shown in Figure 15 is a histogram of the y values (the dimension toward the rear wall) of all the points in the filled in boundary. The short peak at the left of the histogram corresponds to points on the cone and on the box in front of the can. The large peak is the front surface of the can, then the curve falls slowly as range values follow the curve of the cylinder toward the rear. The curve begins to rise as we explore the back wall of the can, then we reach a pronounced peak at the rear wall.

The cursor is used again to select the portions of the histogram that correspond to the portion of the scene we are interested in. The two crosses in the bottom of Figure 15 show the range selected. All the points of the original mask that are out of the range selected are deleted, resulting in the mask shown in Figure 16.



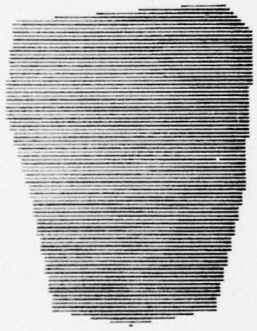
SA-1187-36

FIGURE 12 MASK A: JUMP BOUNDARIES



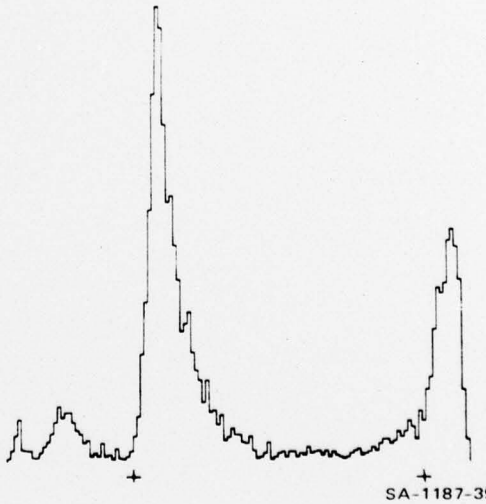
SA-1187-37

FIGURE 13 OUTLINING A SHAPE



SA-1187-38

FIGURE 14 MASK B: OUTLINED AREA



SA-1187-39

FIGURE 15 HISTOGRAM OF y VALUES

SA-1187-40

FIGURE 16 MASK C: POINTS SELECTED FROM HISTOGRAM

SA-1187-41

FIGURE 17 RECTANGLE OUTLINED WITH CURSOR

SA-1187-42

FIGURE 18 MASK D: GENERATED FROM OUTLINE

SA-1187-43

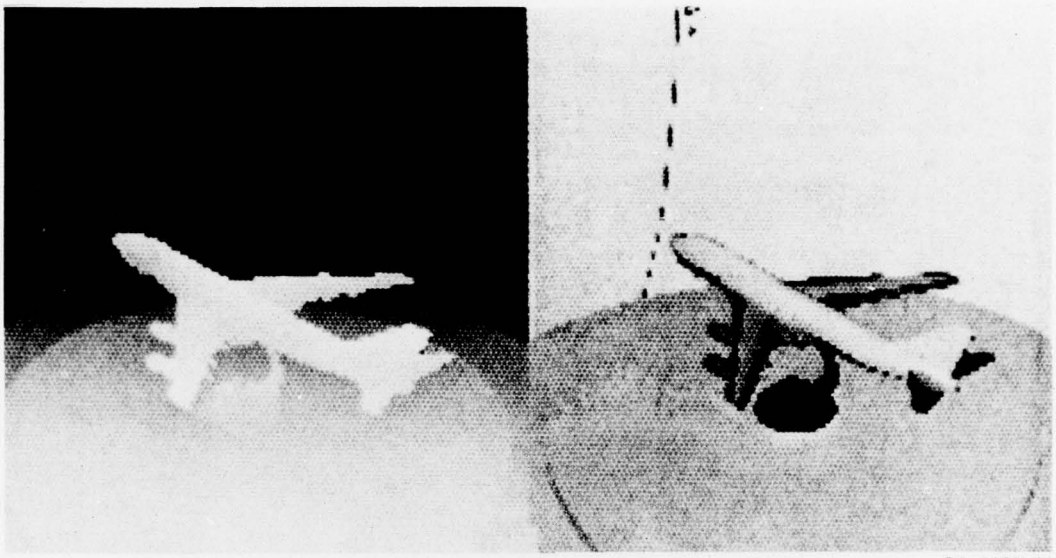
FIGURE 19 MASK E: MASKS C AND D ORED

Because the separation between the can and the wall is insufficient to distinguish between the can and the wall on the basis of y values alone, a portion of the rear of the can is missing from the mask. This may be fixed by drawing a small rectangle around the portion left out, and ORing the rectangle with the previous mask to produce a filled in mask as shown in Figures 17 through 19.

A different example illustrates the use of connectivity and grow/shrink operations. Figure 20 shows the depth and intensity pictures of a model airplane standing on a circular table. The jump boundaries of Figure 21 delineate the airplane well, making it feasible to use connectivity to isolate the airplane from the scene. After the mask is inverted by the NOT operation, as shown in Figure 22, the connectivity test detects three connected areas in the scene, separated by the jump boundary lines. By indicating the area we want with a cursor, we isolate the interior of the jump boundary lines, as shown in Figure 23.

The number of points in the isolated image is small; we would have trouble fitting cylinders and cones to the points we have, particularly on the wings. There are many data points lying on the airplane that are not included in the result so far, simply because they are part of a jump boundary, i.e., they have neighbors at a different range from themselves. It would be desirable to include these otherwise valid points in the data.

This is accomplished by a two-step process involving growing and histogramming. First the mask is grown, or expanded, two pixels. Every bit adjacent to a one, or adjacent but one to a one becomes a one. The grown outline is shown in Figure 24. Then a histogram of z-values (heights) is created, as shown in Figure 25. Various peaks in the histogram correspond to a portion of the rear wall, the table top, and the airplane. The airplane may be isolated on the basis of its z-values, with the result as shown in Figure 26.

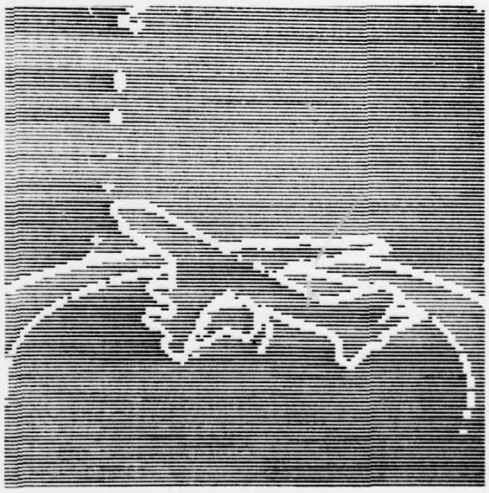


SA-1187-44

FIGURE 20 RANGE AND INTENSITY IMAGES FOR AIRPLANE SCENE

SA-1187-45

FIGURE 21 MASK A: JUMP BOUNDARIES



SA-1187-46

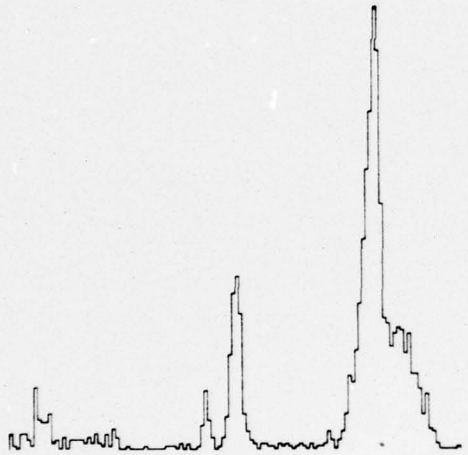
FIGURE 22 MASK B: INVERSE OF MASK A

SA-1187-47

FIGURE 23 MASK C: ISOLATED BY CONNECTIVITY ANALYSIS

SA-1187-48

FIGURE 24 MASK D: GROWING



SA-1187-49

FIGURE 25 HISTOGRAM OF z VALUES

SA-1187-50

FIGURE 26 MASK E: POINTS SELECTED FROM HISTOGRAM

It is our intention that the techniques demonstrated above, and others of a similar nature, will ultimately be available to an automatic procedure that will perform isolation, segmentation, fitting, and recognition. But in the current state of development, human guidance is needed to sequence and check for correct operation of the techniques.

C. Surface Fitting

In our bottom-up analysis of range data the next step after segmentation is fitting of primitive surfaces to the data. We have a method of fitting cones, cylinders, and rectangular solids (bricks) to patches of range data, that is extremely flexible in allowing constraints to be applied. This flexibility allows a knowledge-based approach to the fitting of models.

The method makes use of iterative hill-climbing to minimize an error function that characterizes the goodness of fit of a surface to the data. The optimization is done by the same steepest-descent subroutine as is used for calibration of the laser range finder.

The choice of variables over which to optimize is important. There are many ways of expressing the equation of a surface, each involving a number of parameters. For example, consider two alternative equations for the cross section of an ellipse:

$$A x^2 + B xy + C y^2 + D x + E y + F = 0 \qquad (1)$$

versus

$$\frac{x'^2}{M^2} + \frac{y'^2}{(k M)^2} = 1 \qquad (2)$$

where

$$x' = (x - X_0) \cos \phi + (y - Y_0) \sin o \;|$$

and

$$y' = - (x - X_0) \sin \phi + (y - Y_0) \cos o \;|$$

Equation (1) is attractive as an error measure because, for efficiency, it is amenable to methods that use summations of x y, and their products. However, the parameters of the equation, A through F, have no relationship to information we might already know about the ellipse.

The parameters of Equation (2). M k, X_0, Y_0, and ϕ, represent size shape (flatness), x-position, y-position, and orientation, respectively.

Frequently, information is available to constrain the fit of a surface to a number of three dimensional points. We might know, for example, the ratio of major to minor dimensions of the ellipse, or we may know that the center of the ellipse must be at a particular place. The finding of a surface that fits a particular set of constraints is simplified if the parameters of the fit are related to the kinds of constraints that are expected Equation (2) is more useful in this respect than Equation (1). Use of this description will permit a knowledge synthesis knowledge obtained elsewhere may be applied to the fitting of a surface.

The hill-climbing method of fitting a surface to a set of range points requires an error-measuring function that can quantify the nearness of a hypothetical surface to the points. Then we can optimize over the parameters that describe the surface, to minimize the error measure.

Our error measuring function builds a homogeneous transform, based on the position and orientation parameters of the surface, that will make the axis of the transformed surface vertical, centered on the origin of transformed coordinates The error measure is the mean-square distance of all the range data points from the surface. In the case of a rectangular cross section, a small number of simple comparisons determines which flat surface is nearest a given point, and the distance is easily measured.

The exact distance from a point (x, y, z) to a right circular cylinder of radius M, centered on the z-axis, is given by

$$\text{distance} = \sqrt{x^2 + y^2} - M$$

Because the square root operation is computationally expensive, the distance is approximated by the formula

$$D' = 1/2 \frac{x^2 + y'^2 - M^2}{M}$$

which has the same magnitude and gradient in the vicinity of the surface.

When the cross section is elliptical with semimajor axis M and semiminor axis k M, the distance approximation is replaced by

$$D' = 1/2 \; \frac{x^2 + y'^2 - M^2}{M \sqrt{f}}$$

where

$$y' = y / k$$

and

$$f = \frac{1 + (y'/kx)^2}{1 + (y'/x)^2}$$

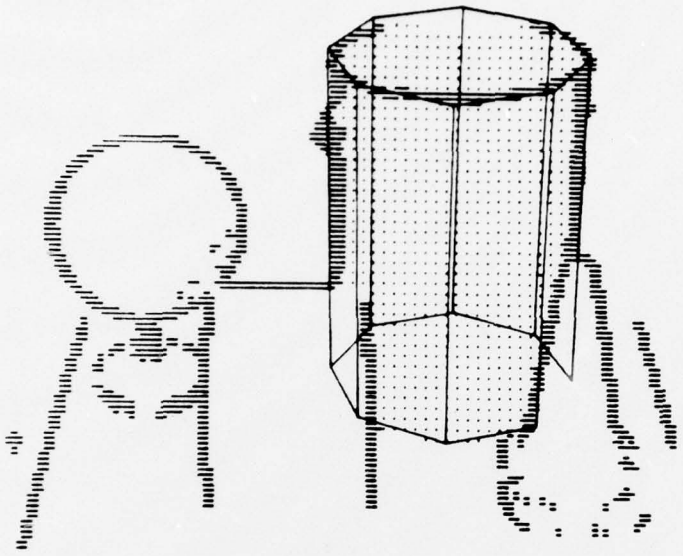
Because D' will be squared before adding it to the running total for all points, the square root need not be evaluated. Transforming y into y' is accomplished "for free" by multiplying the second row of the homogeneous transform by k.

In the case of a cone with linearly varying dimension, the parameter M is a linear function of z. In the equations for D' above replace M with the product M_0 (1-c) z, where M_0 is the semimajor dimension at the x-y plane and c is the "conicity." This function is calculated "for free" by making use of the otherwise unused third row of the homogeneous transform.

This method is fast, requiring one matrix multiplication and eight to fifteen additional arithmetic operations per point Additionally, the method avoids most of the degeneracies and biases inherent in eigenvalue solutions when large scatters of data points are encountered [16].

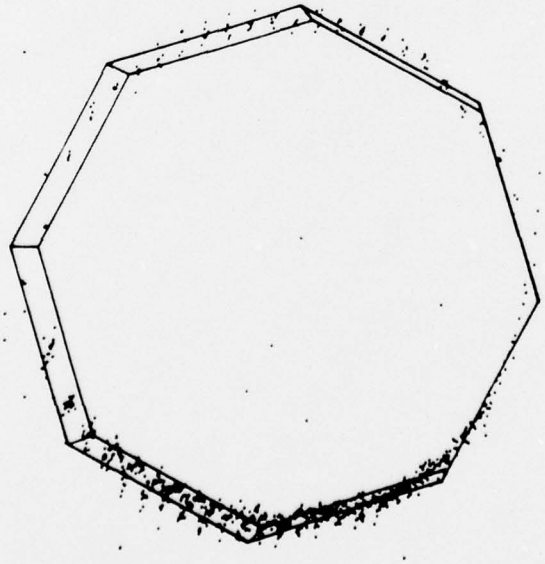
An interactive procedure guides the fitting. The user may use the display cursor to indicate an initial estimate of where the axis lies. The user then supplies an initial guess as to the diameter, specifies constraints, and initiates the hill-climbing optimization.

Surface fitting provides no information as to the length of a primitive generalized cylinder, or the location of its ends. Once the



SA-1187-51

FIGURE 27 FITTED CYLINDER AND JUMP BOUNDARIES



SA-1187-52

FIGURE 28 TOP VIEW OF FITTED CYLINDER AND RANGE DATA

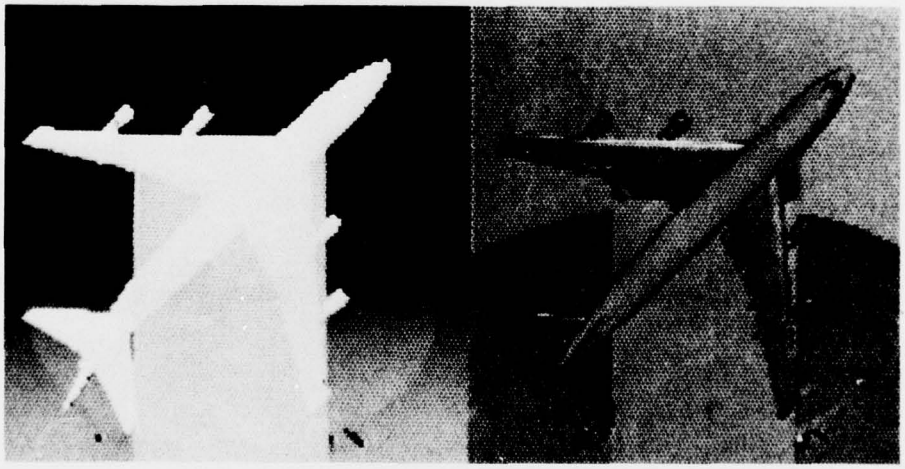
optimally fitting cylindrical surface has been found, we use the extreme values of z in transformed coordinates (parallel to the axis) to locate the ends and to measure the cylinder's length.

The cylinder thus found is shown superimposed on the original jump boundaries in Figure 27. Figure 28 is a three-dimensional perspective display of the points selected for the waste container, from a point of view above the top, with the outline of the fitted cylinder superimposed

D. Interactive Model Building

For display purposes, models of complex objects may be built up by using interactive means to segment an object into subparts. and to model each subpart by fitting the appropriate primitive surface.

For example, consider the scene shown in Figure 29. Here, a model airplane was attached to a wooden board for support in order to obtain with the range finder a nearly perpendicular view of the model Figure 29 displays the range and intensity images obtained.



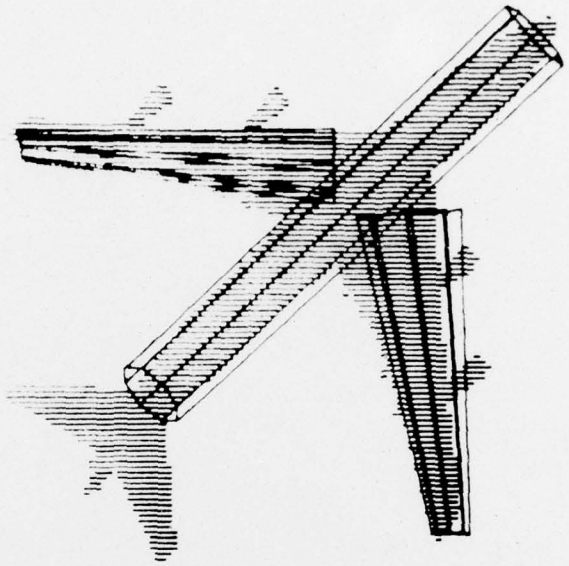
SA-1187-53

FIGURE 29 RANGE AND INTENSITY IMAGES FOR MODEL AIRPLANE SCENE

Portions corresponding to the fuselage and each wing were segmented out from the image. Flattened cones were used to model the wings (The minor diameter was constrained to be one-tenth the major diameter.) The fuselage was modeled with a circular cylinder. The results are shown in Figure 30 superimposed on the silhouette of the entire airplane Figure 31 shows the fuselage and wings in perspective from an angle other than the one from which the picture was taken. The point of view is similar to the one used to make Figure 2.

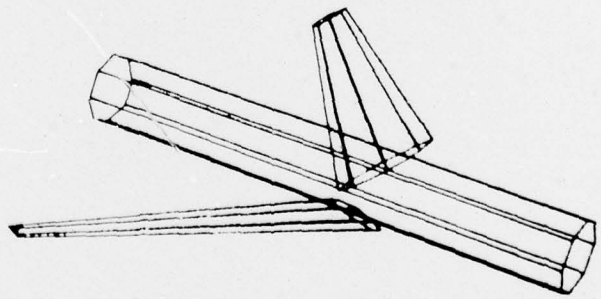
There are many ways these methods could be made more effective, more automatic and more useful. The procedures are bottom-up and manually guided. More effective use could be made of boundary information. There is currently only a very minimal connection between the primitives extracted from the range image and the high-level data base used for modeling prototypes and specific objects.

The corrections for these deficiencies are in many cases obvious, but much research remains to be done before a truly general capability for inferring shape from range data is implemented.



SA-1187-54

FIGURE 30 PRIMITIVE SURFACES FIT TO RANGE DATA



SA-1187-55

FIGURE 31 PRIMITIVE SURFACES FIT TO RANGE DATA: ALTERNATE VIEWPOINT

REFERENCES

1. R. E. Barnhill et al., "Smooth Interpolation in Triangles," Journal of Approximation Theory, Vol. 8, pp. 114-128 (1973).

2. S A Coons, "Surfaces for Computer-Aided Design of Space Forms," Project MAC Report MAC-TR-41. Massachusetts Institute of Technology, Cambridge, Massachusetts (June 1967).

3. I C Braid, Designing with Volumes (Cantab Press, Cambridge, England 1973).

4. B G. Baumgart "GEOMED--A Geometric Editor." Stanford Artificial Intelligence Project Memo AIM-232, Stanford University, Stanford. California (May 1974).

5. A. A. G. Requicha, "Part and Assembly Description Languages -- I. Dimensioning and Tolerancing Facilities in PADL," Production Automation Project Report TM-19, University of Rochester, Rochester, New York (1976).

6. A. A. G. Requicha et al., "Part and Assembly Description Languages -- II, Proposed Specifications for Definitional Facilities in PADL-1.n and Tentative Specifications for Command Facilities," Production Automation Project Report TM-20a, University of Rochester, Rochester New York (1974).

7. N. Okino et al., "TIPS-1: Technical Information Processing System for Computer-Aided Design," in Computer Languages for Numerical Control (American Elsevier, New York, 1973).

8. G. J. Agin and T. O. Binford, "Computer Descriptions of Curved Objects," IEEE Transactions on Computers, Vol. 25, No. 4 (April 1976).

9. R. K. Nevatia and T. O Binford, "Structured Descriptions of Complex Objects," Proc. Third International Joint Conference on Artificial Intelligence, Stanford, California (1973).

10. J. M. Hollerbach, "Hierarchical Shape Description of Objects by Selection and Modification of Prototypes," Master's Thesis Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts (1975).

11. Clark Weissman, LISP 1.5 Primer (Dickenson Publishing Company, Belmont, California, 1967).

12. G. J. Agin, "Hierarchical Representation of Three-Dimensional Objects," Annual Report, Contract N00014-71-C-0294, SRI Project 1187, Stanford Research Institute, Menlo Park, California (December 1975).

13. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis (John Wiley and Sons, New York, New York, 1973).

14. D. Nitzan, A. E. Brain, and R. O Duda, "The Measurement and Use of Registered Reflectance and Range Data in Scene Analysis," Proc. IEEE, Vol. 65,, No. 2 (February 1977).

15. I. Sobel, "Camera Models and Machine Perception," Stanford Artificial Intelligence Laboratory Memo AIM-121, Stanford University, Stanford, California (May 1970).

16. G. J. Agin, "Representation and Description of Curved Objects," Stanford Artificial Intelligence Project Memo AIM-173, Computer Science Department, Stanford University, Stanford, California (October 1972).